CODE PROCESSING METHOD AND APPARATUS, COMPUTE DEVICE CLUSTER, AND STORAGE MEDIUM

[0001] This application claims priority to Chinese Patent Application No. 202410204960.3, filed on February 23, 2024 and entitled "CODE PROCESSING METHOD AND APPARATUS, COMPUTE DEVICE CLUSTER, AND STORAGE MEDIUM", which is incorporated herein by reference in its entirety.

TECHNICAL FIELD

[0002] This application relates to the field of artificial intelligence technologies, and in particular, to a code processing method and apparatus, a compute device cluster, and a storage medium.

BACKGROUND

[0003] The artificial intelligence technology can be applied to the field of code processing. In a code processing process, modification suggestions can be proposed to a user based on original code of the user. The modification suggestions typically include a code segment and a descriptive text. The descriptive text indicates a manner of processing the original code. The processing manner may be inserting the code segment in the modification suggestions into the original code, deleting the code segment in the modification suggestions from the original code, or replacing some code in the original code with the code segment in the modification suggestions. The user processes the original code according to the modification suggestions.

[0004] Currently, a code processing method is to provide a one-click insertion function for the user, and the one-click insertion function provides a button for the user. In response to a clicking operation on the button by the user, the code segment in the modification suggestions is inserted into a location where a cursor is located; or a selected original code is replaced with a code segment in the modification suggestions.

[0005] However, the foregoing code processing method requires manual understanding of the descriptive text in the modification suggestions, and manual identification of a modification location, in the original code, corresponding to the code segment in the modification suggestions. This leads to a high error rate in the code processing process and low code processing efficiency.

SUMMARY

[0006] Embodiments of this application provide a code processing method and apparatus, a compute device cluster, and a storage medium, to reduce an error rate in a code processing process and improve code processing efficiency. The technical solutions are as follows.

[0007] According to a first aspect, a code processing method is provided. The method includes: A code development client obtains first code obtained through development. The code development client analyzes the obtained first code in response to a code analysis instruction, to obtain modification suggestion information for the first code, where the modification suggestion information includes a descriptive text and a to-be-processed code segment, the descriptive text indicates a modification intent for the first code based on the code segment, and the modification intent includes deleting the code segment from the first code, adding the code segment to the first code, and replacing a code segment in the first code with the code segment. The code development client performs analysis based on the modification suggestion information obtained through analysis and the first code, to obtain the modification intent and a code modification location, in the first code, that corresponds to the modification suggestion information. The code development client performs, at the code modification location in the first code based on the modification intent and the code modification location, processing corresponding to the modification intent based on the code segment.

[0008] The descriptive text in the modification suggestion information is a modification suggestion, generated by a text generation model, that is described in a natural language, and may indicate an error cause of the first code, a location at which an error occurs, a modification method for the error, and the like. The code segment in the modification suggestion information corresponds to the descriptive text. The code segment may be complete and modified first code, or may be a modification result at the code modification location in the first code. The modification intent for the first code includes: inserting the code segment in the modification suggestion information, deleting the code segment in the modification suggestion information, modifying a part of code in the first code to the code segment in the modification suggestion information, and the like, which may be briefly referred to as adding, deleting, and modifying. The code modification location may be represented by a line number or a segment number.

[0009] According to the code processing method provided in this embodiment of this application, in a code processing process, the first code and the modification suggestion information are obtained. Because there is a specific rule between semantics of the modification suggestion information, an association between the first code and the modification suggestion information, and the like, and the modification intention and the code modification location, the

modification intention and the code modification location can be obtained based on the first code and the modification suggestion information, to implement automated understanding of the modification suggestion information. The first code is processed based on the modification intent and the code modification location. Because an amount of to-be-processed code may be huge, compared with manually understanding the foregoing modification suggestion information and then processing the code, the code processing method provided in this embodiment of this application can reduce an error rate in the code processing process, and improve code processing efficiency.

[0010] Optionally, that the code development client performs analysis based on the modification suggestion information obtained through analysis and the first code, to obtain the modification intent and the code modification location, in the first code, that corresponds to the modification suggestion information includes: In response to a trigger operation for the modification suggestion information, the code development client performs analysis based on the modification suggestion information and the first code, to obtain the modification intent and the code modification location. A subsequent code processing process can be performed only in response to the trigger operation for the modification suggestion information, so that accuracy of the modification suggestion information used in the subsequent code processing process can be ensured, thereby improving accuracy of code processing.

[0011] Optionally, that the code development client performs, at the code modification location in the first code based on the modification intent and the code modification location, processing corresponding to the modification intent based on the code segment includes: The code development client performs, at the code modification location in the first code based on the modification intent and the code modification location, processing corresponding to the modification intent based on the code segment, to obtain second code. The code development client displays the second code in a preview window. If an approval instruction is received, the code development client replaces the first code with the second code. After the modification intent and the code modification location are obtained, the first code is not automatically modified, but a modification result is provided for a user to preview. After the user gives approval, a modified code is used to replace the original code. This can avoid a code processing error caused by a recognition error of the modification intent or the code modification location.

[0012] Optionally, that the code development client performs, at the code modification location in the first code based on the modification intent and the code modification location, processing corresponding to the modification intent based on the code segment, to obtain the second code includes: If the modification intent is adding, the code development client inserts the code segment in the modification suggestion information after code at the code modification

location in the first code, to obtain the second code; if the modification intent is deleting, the code development client deletes code at the code modification location from the first code, to obtain the second code; or if the modification intent is modifying, the code development client deletes code at the code modification location from the first code, and inserts the code segment in the modification suggestion information at the code modification location, to obtain the second code.

[0013] Optionally, the modification intent and the code modification location are obtained by using a code processing model, and a training process of the code processing model includes: obtaining a first training dataset, where the first training dataset includes a plurality of groups of first sample data and first label, the first sample data includes one group of first code and modification suggestion information, and the first label includes one group of modification intent and code modification location corresponding to the first sample data; and training the code processing model based on the first training dataset. The code processing model has a powerful natural language processing capability, and can generate both the modification intent and the code modification location, so that efficiency is relatively high.

[0014] Optionally, the modification intent is obtained by using an intent recognition model, and a training process of the intent recognition model includes: obtaining a second training dataset, where the second training dataset includes a plurality of groups of second sample data and second label, the second sample data includes one group of first code and modification suggestion information, and the second label includes one modification intent corresponding to the second sample data; and training the intent recognition model based on the second training dataset.

[0015] Optionally, the code modification location is obtained by using a locating model, and a training process of the locating model includes: obtaining a third training dataset, where the third training dataset includes a plurality of groups of third sample data and third label, the third sample data includes one group of first code, modification suggestion information, and modification intent, and the third label includes one code modification location corresponding to the third sample data; and training the locating model based on the third training dataset.

[0016] The intent recognition model and the locating model are natural language processing (natural language processing, NLP) models that can be used for text generation or classification, for example, a conventional machine learning model (such as Naive Bayes classification) or a deep learning model (a convolutional neural network model, a recurrent neural network model, and a transformer (Transformer) model).

[0017] According to a second aspect, a code processing apparatus is provided, and is configured to perform the foregoing code processing method. Specifically, the code processing apparatus includes a functional module configured to perform the code processing method provided in any one of the first aspect or the optional manners of the first aspect.

[0018] According to a third aspect, a compute device cluster is provided, including at least one compute device. Each compute device includes a processor and a memory. The processor of the at least one compute device is configured to execute instructions stored in the memory of the at least one compute device, so that the compute device cluster performs an operation to implement the code processing method provided in any one of the first aspect or the optional manners of the first aspect.

[0019] According to a fourth aspect, a computer-readable storage medium is provided, including computer program instructions. When the computer program instructions are executed by a compute device cluster, the compute device cluster performs an operation performed in the code processing method provided in any one of the first aspect or the optional manners of the first aspect.

[0020] According to a fifth aspect, a computer program product including instructions is provided. When the instructions are run by a compute device cluster, the compute device cluster is caused to perform the code processing method provided in any one of the first aspect or the optional implementations of any one of the foregoing aspects.

[0021] In this application, the implementations provided in the foregoing aspects may be further combined to provide more implementations.

BRIEF DESCRIPTION OF DRAWINGS

[0022] FIG. 1 is a diagram of an implementation environment of a code processing method according to an embodiment of this application;

[0023] FIG. 2 is a diagram of a structure of a code processing system according to an embodiment of this application;

[0024] FIG. 3 is a diagram of a structure of a compute device according to an embodiment of this application;

[0025] FIG. 4 is a diagram of a structure of a compute device cluster according to an embodiment of this application;

[0026] FIG. 5 is a diagram of a structure of another compute device cluster according to an embodiment of this application;

[0027] FIG. 6 is a brief flowchart of a code processing method according to an embodiment of this application;

[0028] FIG. 7A and FIG. 7B are a flowchart of a code processing method according to an embodiment of this application;

[0029] FIG. 8 is a diagram of modification suggestion information according to an

embodiment of this application;

[0030] FIG. 9 is a flowchart of a code processing model training method according to an embodiment of this application;

[0031] FIG. 10 is a brief flowchart of another code processing method according to an embodiment of this application;

[0032] FIG. 11A and FIG. 11B are a flowchart of another code processing method according to an embodiment of this application;

[0033] FIG. 12 is a flowchart of an intent recognition model training method according to an embodiment of this application;

[0034] FIG. 13 is a flowchart of a locating model training method according to an embodiment of this application; and

[0035] FIG. 14 is a diagram of a structure of a code processing apparatus according to an embodiment of this application.

DESCRIPTION OF EMBODIMENTS

[0036] To make the objectives, technical solutions, and advantages of this application clearer, the following further describes the implementations of this application in detail with reference to the accompanying drawings.

[0037] FIG. 1 is a diagram of an implementation environment of a code processing method according to an embodiment of this application. As shown in FIG. 1, the implementation environment includes a terminal 101 and a server 102. The terminal 101 and the server 102 are communicatively connected via a wired network or a wireless network. Quantities of terminals and servers shown in FIG. 1 are merely examples, and the quantities of terminals and servers in the implementation environment are not limited in this embodiment of this application.

[0038] The terminal 101 is configured to: as a code development client, obtain first code, and send the first code to the server 102. The terminal 101 includes but is not limited to a scanner, a smartphone, a smart television, a television box, a keyboard, a mouse, a game controller, an intelligent wearable device, a tablet computer, a portable personal computer, a mobile Internet device, and the like. A type of the terminal 101 is not limited in this embodiment of this application.

The server 102 is configured to: process the first code according to the code processing method provided in embodiments of this application, to obtain a modification intent and a code modification location, and send the modification intent and the code modification location to the terminal 101. The server 102 may be an independent physical server, or may be a server cluster or a distributed system including a plurality of physical servers, or may be a cloud server that provides

basic cloud computing services such as a cloud service, a cloud database, cloud computing, a cloud function, cloud storage, a network service, cloud communication, a middleware service, a domain name service, a security service, a content delivery network (content delivery network, CDN), big data, and an artificial intelligence platform.

[0039] In this embodiment of this application, the terminal 101 obtains the first code, and sends the first code to the server 102; and the server 102 processes the first code. In some embodiments, the terminal 101 or the server 102 can also independently complete a process of obtaining the first code and processing the first code.

[0040] The foregoing code processing method may be implemented by using a code processing system. As shown in FIG. 2, the code processing system includes a code processing apparatus 200. The code processing apparatus 200 includes a modification suggestion obtaining module 201 and a code intelligent processing module 202, configured to interact with a server to obtain a modification intent and a code modification location. In a code processing process, the modification suggestion obtaining module 201 interacts with the server, to obtain modification suggestion information output by a suggestion obtaining model on the server for the first code. The code intelligent processing module 202 interacts with the server based on the modification suggestion information and the first code, to obtain the modification intent and the code modification location that are output by a code processing model (or an intent recognition model and a locating model) on the server for the modification suggestion information. The code processing apparatus 200 may further process the first code based on the obtained modification intent and the obtained code modification location. It should be noted that the foregoing module division is merely an example, and this embodiment of this application is not limited thereto. The suggestion obtaining model may be a large language model (large language model, LLM), or may be another natural language processing model. The LLM is a deep learning model with a large scale and a large quantity of parameters. Language structures, syntax knowledge, and semantics can be learned from a large-scale dataset to understand and generate human languages. The code processing model is a model with a powerful natural language processing capability, such as the LLM. The intent recognition model may be an NLP model that can be used for text generation or classification, for example, a conventional machine learning model (such as Naive Bayes classification) or a deep learning model (such as a convolutional neural network model, a recurrent neural network model, and a transformer). This is not limited in this embodiment of this application.

[0041] For example, the code processing apparatus 200 is a programming plug-in in an integrated development environment (integrated development environment, IDE) client. During code debugging, the programming plug-in interacts with the server to obtain the modification intent and the modification location of code. The modification intent is a code processing manner,

for example, adding several lines of code to the code, deleting several lines of code from the code, or replacing several lines of code in the code. The code modification location is a line number of code corresponding to the modification intent.

[0042] The code processing apparatus 200 may be implemented through software, or may be implemented through hardware. For example, the following describes an implementation of the code processing apparatus 200.

[0043] As an example of a software functional unit, the code processing apparatus 200 may include code running on a computing instance. The computing instance may be at least one of the following compute devices: a physical host (compute device), a virtual machine, a container, and the like. Further, there may be one or more compute devices. For example, the code processing apparatus 200 may include code running on a plurality of hosts/virtual machines/containers. It should be noted that, the plurality of hosts/virtual machines/containers configured to run the application program may be distributed in a same region (region), or may be distributed in different regions. The plurality of hosts/virtual machines/containers configured to run the code may be distributed in a same availability zone (availability zone, AZ), or may be distributed in different AZs. Each AZ includes one data center or a plurality of data centers with close geographical locations. Usually, one region may include a plurality of AZs.

[0044] Similarly, the plurality of hosts/virtual machines/containers configured to run the code may be distributed on a same virtual private cloud (virtual private cloud, VPC), or may be distributed on a plurality of VPCs. Usually, one VPC is set in one region. A communication gateway needs to be set in each VPC for communication between two VPCs in a same region or between VPCs in different regions. Interconnection between the VPCs is implemented through the communication gateway.

[0045] As an example of a hardware functional unit, the code processing apparatus 200 may include at least one compute device, for example, a server. Alternatively, the code processing apparatus 200 may be a device implemented through an application-specific integrated circuit (application-specific integrated circuit, ASIC), a programmable logic device (programmable logic device, PLD), or the like. The PLD may be implemented through a complex programmable logic device (complex programmable logic device, CPLD), a field-programmable gate array (field programmable gate array, FPGA), generic array logic (generic array logic, GAL), or any combination thereof.

[0046] A plurality of compute devices included in the code processing apparatus 200 may be distributed in a same region, or may be distributed in different regions. The plurality of compute devices included in the code processing apparatus 200 may be distributed in a same AZ, or may be distributed in different AZs. Similarly, the plurality of compute devices included in the code

processing apparatus 200 may be distributed in a same VPC, or may be distributed in a plurality of VPCs. The plurality of compute devices may be any combination of compute devices such as a server, an ASIC, a PLD, a CPLD, an FPGA, and GAL.

[0047] FIG. 3 is a diagram of a structure of a compute device according to an embodiment of this application. The compute device 300 may be the terminal 101 or the server 102 in the foregoing implementation environment. This is not limited in this embodiment of this application. It should be understood that the compute device described below may implement any function in any one of the following methods.

[0048] As shown in FIG. 3, the compute device 300 includes a bus 302, a processor 304, a memory 306, and a communication interface 308. The processor 304, the memory 306, and the communication interface 308 communicate with each other through the bus 302. It should be understood that quantities of processors and memories in the compute device 300 are not limited in this application.

[0049] The bus 302 may be a peripheral component interconnect (peripheral component interconnect, PCI) bus, an extended industry standard architecture (extended industry standard architecture, EISA) bus, or the like. Buses may be classified into an address bus, a data bus, a control bus, and the like. For ease of representation, only one line is used for representation in FIG. 3, but this does not mean that there is only one bus or only one type of bus. The bus 302 may include a path for transmitting information between components (for example, the memory 306, the processor 304, and the communication interface 308) of the compute device 300.

[0050] The processor 304 may include any one or more of processors such as a central processing unit (central processing unit, CPU), a graphics processing unit (graphics processing unit, GPU), a microprocessor (microprocessor, MP), or a digital signal processor (digital signal processor, DSP).

[0051] The memory 306 may include a volatile memory (volatile memory), for example, a random access memory (random access memory, RAM). The processor 304 may further include a non-volatile memory (non-volatile memory), for example, a read-only memory (read-only memory, ROM), a flash memory, a hard disk drive (hard disk drive, HDD), or a solid state drive (solid state drive, SSD).

[0052] The memory 306 stores executable program code, and the processor 304 executes the executable program code to implement a function of the code processing apparatus 200, so as to implement the code processing method. In other words, the memory 306 stores instructions used to perform the code processing method.

[0053] The communication interface 308 uses a transceiver module, for example, but not limited to, a network interface card or a transceiver, to implement communication between the

compute device 300 and another device or a communication network.

[0054] An embodiment of this application further provides a compute device cluster. The compute device cluster includes at least one compute device. The compute device may be a server, for example, a central server, an edge server, or a local server in a local data center. In some embodiments, the compute device may alternatively be a terminal device, for example, a desktop computer, a notebook computer, or a smartphone.

[0055] As shown in FIG. 4, the compute device cluster includes at least one compute device 300. Memories 306 in one or more compute devices 300 in the compute device cluster may store a same instruction used to perform the code processing method.

[0056] In some possible implementations, a memory 306 of each of the one or more compute devices 300 in the compute device cluster may store some instructions used to perform the code processing method. In other words, a combination of the one or more compute devices 300 may jointly execute instructions for performing the code processing method.

[0057] It should be noted that memories 306 in different compute devices 300 in the compute device cluster may store different instructions, which are respectively used to perform some functions of the code processing apparatus 200. In other words, the instructions stored in the memories 306 in the different compute devices 300 may implement functions of one or more of the modification suggestion obtaining module 201 and the code intelligent processing module 202.

[0058] In some possible implementations, the one or more compute devices in the compute device cluster may be connected via a network. The network may be a wide area network, a local area network, or the like. FIG. 5 shows a possible implementation. As shown in FIG. 5, two compute devices 300A and 300B are connected via a network. Specifically, each compute device is connected to the network through a communication interface in the compute device. In this type of possible implementation, a memory 306 in the compute device 300A stores instructions for executing a function of the modification suggestion obtaining module 201. In addition, a memory 306 in the compute device 300B stores instructions for executing a function of the code intelligent processing module 202.

[0059] In a connection manner between compute device clusters shown in FIG. 5, it may be considered that the code processing method provided in this application needs to store a large amount of data. Therefore, it is considered that a function implemented by the code intelligent processing module 202 is performed by the compute device 300B.

[0060] It should be understood that functions of the compute device 300A shown in FIG. 5 may alternatively be completed by a plurality of compute devices 300. Similarly, functions of the compute device 300B may alternatively be completed by a plurality of compute devices 300.

[0061] In related technologies, descriptive text is not obtained based on a unified description

template, and a code segment in modification suggestion information is not obtained according to a unified modification method. Therefore, the provided modification suggestion information is random and diversified, and is relatively vague. The modification suggestion information cannot clearly indicate that modification intent is adding, deleting, or modifying, and cannot clearly indicate a specific code modification location either. As a result, a server cannot recognize the modification suggestion information by using a fixed recognition template, to obtain the modification intent or the code modification location. Currently, the modification suggestion information is usually manually understood to obtain the modification intent and the code modification location, and then a compute device is indicated to process first code. When a large amount of first code is processed, manual understanding is time-consuming and error-prone. Consequently, an error rate of code processing is relatively high, and code processing efficiency is relatively low.

[0062] To resolve the foregoing problems, embodiments of this application provide a code processing method. In the method, a pre-trained model is invoked to process the first code and the modification suggestion information, to obtain the modification intent and the code modification location, so as to understand the modification suggestion information. The model is trained to learn structures and semantics of the first code and the modification suggestion information, an association between the first code and the modification suggestion information, and the like, so that human thinking can be simulated to accurately understand the modification suggestion information. The model is used to replace manual operations, so that the error rate of code processing can be reduced, and code processing efficiency can be improved.

[0063] Embodiments of this application provide two implementations of understanding the modification suggestion information by using the model. Implementation 1: One model is used in the code processing process, that is, a code processing model. The model can output the modification intent and the code modification location based on the first code and the modification suggestion information. Implementation 2: Two models are used in the code processing process, that is, an intent recognition model and a locating model. The intent recognition model can output the modification intent based on the modification suggestion information for the first code, and the locating model can output the code modification location based on the first code, the modification suggestion information, and the modification intent. The following separately describes the foregoing two implementations with reference to FIG. 6 to FIG. 13.

[0064] FIG. 6 is a flowchart of performing code processing based on the code processing model. FIG. 6 includes two processes. One is a process of obtaining the first code and the modification suggestion information; then inputting the first code and the modification suggestion information into the code processing model, to output the modification intent and the code

modification location; and then processing the first code based on the modification intent and the code modification location. The other is a process of training the code processing model based on a training dataset. The following further describes the two processes based on FIG. 7A and FIG. 7B to FIG. 9.

[0065] FIG. 7A and FIG. 7B are a flowchart of a code processing method according to an embodiment of this application. With reference to FIG. 7A and FIG. 7B, the following describes in more detail the code processing method implemented based on a code processing model by using an example in which a code development client obtains first code and a server obtains a modification intent and a code modification location. The method includes the following steps.

[0066] 700: The code development client displays a code processing interface, where the code processing interface is used to provide a code processing function.

[0067] The code processing interface is a function interface provided by the code development client, and includes a plurality of function buttons. Code processing functions corresponding to the function buttons are different, and the function buttons can assist in code processing.

[0068] For example, the code processing interface is a function interface for code debugging, and can be implemented as a programming plug-in of an IDE client, so that a user can open the code processing interface in the IDE client to process code.

[0069] 701: The code development client obtains, in response to an operation performed on the code processing function in the code processing interface, the first code obtained through development.

[0070] The code processing function may provide a plurality of input manners, to obtain the first code. For example, the code processing function may provide an input interface, and the input interface displays different code input manners, for example, inputting the first code or inputting related information of the first code. If the code input manner is inputting the first code, the code development client directly obtains the input first code through the input interface. If the code input manner is inputting the related information of the first code, the related information of the first code is first obtained through the input interface, and then the first code is obtained based on the related information of the first code. The related information of the first code includes an address of the first code, identification information of the first code, and the like. This is not limited in this embodiment of this application.

[0071] For example, the code processing interface is opened in the IDE client, code is input into the input interface provided by the code processing interface, and the IDE client obtains the input code from the input interface. Alternatively, the code processing interface is opened in the IDE client, an address of code is input into the input interface provided by the code processing interface, and the IDE client obtains the address of the code from the input interface, and then

obtains the code from the address.

[0072] 702: The code development client sends a code processing instruction to the server in response to a commit operation on the first code, where the code processing instruction carries the first code.

[0073] In this embodiment of this application, the code processing interface provides a commit button for the user, and the user taps the button to initiate the commit operation. In response to the commit operation, the code development client sends, to the server, the code processing instruction carrying the first code.

[0074] Steps 700 to 702 are described by using an example in which the code processing instruction carries the first code. In some embodiments, the code processing instruction carries the related information of the first code. After the user inputs the related information of the first code into the input interface, the code development client does not obtain the first code based on the related information, but sends, to the server in response to the commit operation on the first code, the code processing instruction carrying the related information of the first code.

[0075] 703: The server receives the code processing instruction.

[0076] In this embodiment of this application, an example in which the server receives the code processing instruction sent by the code development client is used for description. In some embodiments, the server periodically triggers the code processing instruction, to process the first code stored in the server.

[0077] 704: The server obtains the first code based on the code processing instruction.

[0078] In this embodiment of this application, the code processing instruction carries the first code, and the server directly obtains the first code carried in the code processing instruction. In some embodiments, the code processing instruction carries the related information of the first code, and the server obtains the first code based on the related information.

[0079] 705: The server obtains modification suggestion information for the first code based on the first code by using a suggestion obtaining model, and returns the modification suggestion information for the first code to the code development client, where the modification suggestion information includes a descriptive text and a to-be-processed code segment, the descriptive text indicates the modification intent for the first code based on the code segment, and the modification intent includes deleting the code segment from the first code, adding the code segment to the first code, and replacing a code segment in the first code with the code segment.

[0080] In this embodiment of this application, the server invokes the suggestion obtaining model to process the first code, to obtain the modification suggestion information for the first code. When the modification suggestion information is returned to the code development client, the modification suggestion information may be in a form shown in FIG. 8. Refer to FIG. 8. An area

A is a display area of the modification suggestion information, descriptive texts are in areas B1 and B2, and a modified code segment is in an area C. A form of the modification suggestion information is not limited in this embodiment of this application.

[0081] For example, the suggestion obtaining model can check variable settings, data structures, system and standard library calls, redundant code, loop conditions, and the like in the code based on a learned code structure, programming syntax, and the like, find a problem in the code, generate the modification suggestion information for the problem, and return the modification suggestion information to the code development client. The modification suggestion information includes the modified code segment and the descriptive text for the modification. The suggestion obtaining model is stored in the server, and the code development client can enable, by interacting with the server, the suggestion obtaining model in the server to process the code, to output the modification suggestion information.

[0082] In this embodiment of this application, an example in which the server returns the modification suggestion information to the code development client after obtaining the modification suggestion information by using the suggestion obtaining model is used for description. The modification suggestion information is provided for the code development client, to display the modification suggestion information to the user, thereby improving accuracy of code processing. In some embodiments, after obtaining the modification suggestion information, the server directly obtains the modification intent and the code modification location based on the modification suggestion information. In other words, the server can automatically perform a subsequent code processing process, to avoid a problem such as a delay caused by manual check, which has high efficiency.

[0083] The foregoing process is described by using an example in which the server obtains the modification suggestion information by using the suggestion obtaining model. In some embodiments, the server may not obtain the modification suggestion information by using the suggestion obtaining model, but directly receive modification suggestion information that has been generated by another device, in other words, the server receives the first code and the modification suggestion information. The first code and the modification suggestion information may be carried in the code processing instruction sent by the code development client, or may be carried in a code processing instruction sent by the another device. For example, the code development client sends the first code to the another device, and the another device returns the modification suggestion information to the code development client. If the code development client receives an instruction of continuing processing for the modification suggestion information, the code development client sends the code processing instruction carrying the first code and the modification suggestion information to the server. For another example, the code development client sends the first code

to the another device. After obtaining the modification suggestion information based on the first code, the another device directly sends the code processing instruction that carries the first code and the modification suggestion information to the server.

[0084] Steps 702 to 705 are a possible implementation of analyzing the obtained first code in response to a code analysis instruction, to obtain the modification suggestion information for the first code. In this implementation, the modification suggestion information is obtained through interaction between the code development client and the server, which can be able to process large-scale code processing tasks.

[0085] 706: The code development client displays the modification suggestion information for the first code; and sends an instruction of continuing processing to the server in response to a trigger operation for the modification suggestion information, or stops a code processing process in response to a trigger operation for the modification suggestion information.

[0086] In some embodiments, the modification suggestion information for the first code displayed by the code development client may be in the form shown in FIG. 8. During displaying, the code development client not only displays the modification suggestion information, but also displays a button for continuing processing and a button for stopping processing. The button for continuing processing is used to indicate the code development client to send the instruction of continuing processing to the server in response to the trigger operation, and the button for stopping processing is used to indicate the code development client to stop the code processing process in response to the trigger operation.

[0087] In some embodiments, the code development client sends the instruction of continuing processing to the server via the button for continuing processing in response to the trigger operation for the modification suggestion information. Alternatively, the code development client stops the code processing process via the button for stopping processing in response to the trigger operation for the modification suggestion information. Stopping the code processing process may be that the code development client sends an instruction of stopping processing to the server, so that the server no longer executes the subsequent code processing process; or may be that the code development client no longer executes the subsequent code processing process. If the server does not receive the instruction of continuing processing for the modification suggestion information within preset duration, the server does not execute the subsequent code processing process. Alternatively, stopping the code processing process may be that the code development client opens a manual modification interface, displays the first code in the manual modification interface, and receives a modification operation for the first code. This is not limited in this embodiment of this application.

[0088] 707: If the instruction of continuing processing is received, the server obtains, based on the first code and the modification suggestion information for the first code by using the code

processing model, the modification intent for the first code and the code modification location, in the first code, that corresponds to the modification suggestion information.

[0089] In this embodiment of this application, if the instruction of continuing processing is received, the server concatenates the first code and the modification suggestion information in a preset sequence, invokes the code processing model, and processes concatenated first code and modification suggestion information, to obtain the modification intent and the code modification location. The preset order is consistent with a concatenation order of the first code and the modification suggestion information in training data of the code processing model. The code processing model has a language understanding capability and a text generation capability; and can analyze the first code and the modification suggestion information, to understand the modification intent for the first code, find the code modification location, in the first code, that corresponds to the modification suggestion information, and output the modification intent and the code modification location in a fixed format based on a pre-trained output format or learned language knowledge.

[0090] A process of obtaining the modification intent and the code modification location may be: performing text segmentation on the concatenated first code and modification suggestion information to obtain a plurality of words; converting the plurality of words into a plurality of vectors; invoking the code processing model to calculate the plurality of vectors, to obtain a plurality of combinations and a probability of each combination; selecting one combination from the plurality of combinations; and outputting the combination in the fixed format. Each combination includes a pair of modification intent and code modification location, and the probability indicates a probability that an output is a corresponding combination when an input is the first code. Selecting one combination from the plurality of combinations may be selecting a combination with the highest probability, or may be randomly sampling a combination from a plurality of combinations whose probabilities are greater than a probability threshold. The fixed format includes a sequence of the modification intent and the code modification location, a form of the modification intent, a form of the code modification location, and the like. The form of the modification intent may be the modification intent itself, or may be an identifier corresponding to the modification intent. The form of the code modification location may be an array, a list, or the like. Each vector indicates information such as a location of a corresponding word in the first code or the modification suggestion information and a correlation between the corresponding word and another word.

[0091] For example, a line number represents a location of code. The server concatenates the code and the modification suggestion information based on a sequence in which the code is placed before the modification suggestion information. The code processing model is invoked to analyze

the code with a line number and the modification suggestion information for the code to obtain a combination of the modification intent and the line number and the probability of each combination. In the plurality of combinations whose probabilities are greater than the probability threshold, one combination is randomly sampled, and the combination is output in a format of "modification intent: XXX; code modification location: [XX, XX, ...]". For example, the modification intent obtained after the code processing model perform analysis is "deleting", and line numbers corresponding to code modification locations are 21, 22, 23, 24, and 25. In this case, an output of the code processing model is "modification intent: deleting; code modification locations: [21, 22, 23, 24, 25]", which indicates to delete code in lines 21 to 25.

[0092] In this embodiment of this application, an example in which a pair of modification intent and code modification location corresponds to one probability is used for description. In some embodiments, each modification intent corresponds to one probability, and each code modification location also corresponds to one probability. One modification intent and one code modification location are respectively selected from a plurality of modification intents and a plurality of code modification locations, and the modification intent and the code modification location are output in the fixed format.

[0093] Steps 706 and 707 are a possible implementation of performing analysis based on the modification suggestion information obtained through analysis and the first code, to obtain the modification intent and the code modification location, in the first code, that corresponds to the modification suggestion information. In this implementation, the subsequent processing process is performed after the user confirms the modification suggestion information, so that accuracy of the modification suggestion information can be ensured, thereby improving accuracy of code processing.

[0094] 708: The server sends the modification intent and the code modification location of the modification suggestion information for the first code to the code development client.

[0095] In this embodiment of this application, the server interacts with the code development client, and sends, to the code development client, the modification intent and the code modification location that are output by the code processing model. Subsequent code processing work is performed by the code development client, so that workload of the server can be reduced, and load of the server can be reduced. In some embodiments, the server processes the first code based on the modification intent and the code modification location, to obtain second code, and sends the second code to the code development client. In this way, the user can be unaware of the code processing process, and can directly obtain the second code after the first code is input.

[0096] Steps 705 to 708 are a possible implementation in which the server obtains the modification suggestion information and further obtains, based on the modification suggestion

information, the modification intent and the code modification location through interaction with the code development client. In this implementation, after obtaining the modification suggestion information, the server returns the modification suggestion information to the code development client, and the user on the code development client determines whether to perform a next step of obtaining the modification intent and the code modification location. After receiving the instruction of continuing processing triggered by the code development client, the server obtains the modification intent and the code modification location, and performs the subsequent code processing process. This can better comply with a code processing habit of the user, and further improve accuracy of code processing. In some embodiments, the server may not return the modification suggestion information to the code development client, and the user does not need to determine whether to perform code processing based on the modification suggestion information. Instead, after obtaining the modification suggestion information, the server directly obtains the modification intent and the code modification location based on the modification suggestion information, and then returns the modification intent and the code modification location to the code development client in step 708, to perform the subsequent code processing process. This can avoid a problem such as the delay caused by manual check, which has high efficiency.

[0097] 709: The code development client performs, at the code modification location in the first code based on the modification intent and the code modification location, processing corresponding to the modification intent based on the code segment, to obtain the second code.

[0098] In this embodiment of this application, the code development client processes the first code based on the modification intent and the code modification location that are output by the code processing model, to obtain the second code. If the modification intent is adding, the code segment in the modification suggestion information is inserted after code at the code modification location in the first code, to obtain the second code; if the modification intent is deleting, code at the code modification location is deleted from the first code, to obtain the second code; or if the modification intent is modifying, code at the code modification location is deleted from the first code, and the code segment in the modification suggestion information is inserted at the code modification location, to obtain the second code.

[0099] In some embodiments, if the modification intent is adding, the code development client adds a blank line after the code modification location, and inserts the code segment in the modification suggestion information into the blank line. If the modification intent is deleting, the code development client deletes a code line corresponding to the code modification location. If the modification intent is modifying, the code development client deletes code corresponding to the code modification location, and inputs the code segment in the modification suggestion information into the code modification location. Alternatively, the code development client deletes

the code line corresponding to the code modification location, adds a blank line after a cursor location obtained after deletion is performed, and inserts the code segment in the modification suggestion information into the blank line. In the foregoing code modification manner, the code segment in the modification suggestion information is inserted into the newly added blank line. This complies with a programming specification, and can avoid a code syntax error.

[00100] For example, a line number represents the code modification location. If the modification intent output by the code processing model is adding, and the code modification location is the line 21, the code development client adds a blank line, that is, a newly added line 22, after the line 21 in the code, and inserts the code segment in the modification suggestion information into the line 22. If the modification intent output by the code processing model is deleting, and the code modification locations are the lines 21 to 25, the code development client deletes code lines in the lines 21 to 25 from the code. If the modification intent output by the code processing model is modifying, and the code modification locations are the lines 21 to 25, the code development client deletes the code in the lines 21 to 25 from the code, and inserts the code segment in the modification suggestion information into the lines 21 to 25. Alternatively, the code development client deletes the code lines in the lines 21 to 25 from the code. After the deletion, the cursor stays at the end of the line 20, a blank line, that is, a newly added line 21, is added after the line 20, and the code segment in the modification suggestion information is inserted into the line 21.

[00101] 710: The code development client displays the second code in a preview window of the code processing interface.

[00102] In some embodiments, not only the second code is displayed in the preview window, but also an approval button and a rejection button are displayed. The approval button is used to trigger an approval instruction in response to a tap operation, and the rejection button is used to trigger a rejection instruction in response to a tap operation. Certainly, the approval instruction and the rejection instruction may alternatively be triggered in another manner, for example, through a voice instruction. This is not limited in this embodiment of this application.

[00103] In this embodiment of this application, after the second code is obtained, the code development client provides a preview function. The second code is generated for the user to preview, and is modified after being checked, so that impact caused by an error in an output result of the suggestion obtaining model or the code processing model can be reduced, thereby improving accuracy of code processing. Especially for code, there are some special syntax rules of the code. Because these special syntax rules are seldom applied, it is difficult for the suggestion obtaining model or the code processing model to learn related knowledge of these special syntax rules, resulting in a code processing error. The preview function is used, so that it can be determined

whether an error caused by these special syntax rules exists in processed code, thereby ensuring accuracy of code processing.

[00104] In some embodiments, the code processing interface further provides a manual modification option. In response to a trigger operation on the manual modification option, the code development client displays a manual modification interface, displays the second code in the manual modification interface, and receives a modification operation for the second code. After receiving a commit operation for modified second code, the code development client replaces the first code with the modified second code, to implement processing of the first code.

[00105] 711: If the code development client receives an approval instruction for the second code displayed in the preview window, the code development client replaces the first code with the second code.

[00106] In some embodiments, the code development client replaces the first code with the second code in response to the approval instruction triggered for the second code via the approval button. Alternatively, the code development client performs, in response to the rejection instruction triggered for the second code via the rejection button, an operation corresponding to the rejection instruction. The operation corresponding to the rejection instruction may be obtaining feedback information for the second code, and the feedback information of the second code and the modification suggestion information for the first code are sent together to the server. The server performs a subsequent code processing process again based on the feedback information of the second code and the modification suggestion information for the first code. The operation corresponding to the rejection instruction may alternatively be opening the manual modification interface, displaying the first code in the manual modification interface, and receiving the modification operation on the first code.

[00107] Replacing the first code with the second code may be replacing, by the code development client, the first code in the input interface with the second code, replacing the first code with the second code based on the related information of the first code in the input interface, or the like. This is not limited in this embodiment of this application.

[00108] Code replacement manners are different in different scenarios. For example, the code processing method provided in this embodiment of this application may be a code debugging method in a code release process, a code deployment process, and a code test process, and can replace original code with debugged code. If the code development client provides a code release function, code before processing may be code to be released to a code library. Upon receiving the approval instruction, the code development client releases processed code to the code library, to replace the code before processing. The code before processing may alternatively be code already stored in the code library. After receiving the approval instruction, the code development client

replaces the code in the code library with the processed code. If the code development client provides a code deployment function, the code before processing is to-be-deployed code. After receiving the approval instruction, the code development client deploys the processed code to a platform, to replace the code before processing and add functions to the platform. The processed code may alternatively be code that has been deployed on the platform. After receiving the approval instruction, the code development client replaces code at a corresponding location, of a corresponding name, or with a corresponding function on the platform with the processed code. If the code before processing is to-be-tested code, after receiving the approval instruction, the code development client replaces the code before processing with the processed code to perform code testing. Alternatively, the code replacement manner may be as follows: The IDE client runs on the code development client, the IDE client includes an editing area for inputting code, and the code before processing is in the editing area. In a code debugging process, if the code development client receives the approval instruction, the code development client replaces code in the editing area with the processed code. The foregoing process indicates that the code processing method provided in this embodiment of this application can provide corresponding code debugging functions for different programming phases, and can further perform different processing on code in different phases. Therefore, the code processing method is widely applied.

[00109] In this embodiment of this application, an example in which the first code is replaced with the second code is used for description. In some embodiments, after receiving the approval instruction, the code development client does not replace the first code with the second code, but processes the first code based on the modification intent and the modification location that are returned by the server. In this way, when the second code is relatively large, CPU resources occupied by code processing can be reduced.

[00110] Steps 709 to 711 are a possible implementation of processing the first code based on the modification intent and the code modification location. In this implementation, the second code is provided to the user, so that the user can check the second code, thereby ensuring accuracy of code processing. Based on a check result fed back by the user, a new dataset may be selected to train the model used for code processing; or a model structure may be modified for retraining the model, to adjust the model used for code processing, so as to optimize the model and improve accuracy of the model. The foregoing is described by using an example in which the code development client provides the preview function. The preview function can improve accuracy of code processing through user check. In some embodiments, the code development client does not provide the preview function, and directly processes the first code, so that a problem such as the delay caused by manual check can be avoided, and efficiency is relatively high.

[00111] The embodiment shown in FIG. 7A and FIG. 7B is described by using an example in

which the code development client obtains the first code, and the server obtains the modification intent and the code modification location. This scenario is merely an example scenario. In some embodiments, a compute device has a capability of independently completing a code processing task, performs an entire code processing function, and does not need to interact with another device. Therefore, code processing efficiency is relatively high. In some other embodiments, a plurality of compute devices are respectively responsible for some steps in the code processing task, and collaborate to complete the code processing task. Especially when the code processing task is implemented based on a cloud service, cloud servers collaborate to implement user-unaware processing, so that after the user inputs the first code, the second code is directly obtained.

[00112] According to the code processing method provided in this embodiment of this application, in the code processing process, the first code and the modification suggestion information are obtained. Because there is a specific rule between semantics of the modification suggestion information, an association between the first code and the modification suggestion information, and the like, and the modification intention and the code modification location, the modification intention and the code modification location can be obtained based on the first code and the modification suggestion information, to implement automated understanding of the modification suggestion information. The first code is processed based on the modification intent and the code modification location. Because an amount of to-be-processed code may be huge, compared with manually understanding the foregoing modification suggestion information and then processing the code, the code processing method provided in this embodiment of this application can reduce an error rate in the code processing process, and improve code processing efficiency.

[00113] In the code processing method shown in FIG. 7A and FIG. 7B, the modification intent and the code modification location are obtained by using the code processing model. For example, the code processing model is an LLM. A training process of the code processing model includes a pre-training phase and a fine-tuning phase. In the pre-training phase, a language model is trained by using a large amount of training data, to obtain the code processing model. In this case, the code processing model is a general language model, has learned basic language knowledge, and can be used in a plurality of application scenarios. In the fine-tuning phase, the code processing model obtained in the pre-training phase is fine-tuned by using text data in a related field, so that the code processing model learns knowledge in the specific field, and can process a code processing task in the specific field more accurately. FIG. 9 is a flowchart of a code processing model training method according to an embodiment of this application. With reference to FIG. 9, the following uses an example in which a server performs the model training method for description. Step 900 is a pre-training phase, and steps 901 to 904 are a fine-tuning phase. The

method includes the following steps.

[00114] 900: The server obtains a code processing model.

[00115] The code processing model is a language model. The language model is obtained through training by using a large amount of training data, and can learn language modes and representations such as a structure, syntax, and semantics of a language in a training process. In this embodiment of this application, the server obtains a previously downloaded language model, or receives a language model sent by another device; and uses the language model as the code processing model for subsequent training. Fine-tuning is performed based on the language model. It only needs to learn knowledge of a related field based on general language knowledge, and does not need to train a new model from the beginning, thereby saving time and compute resources.

[00116] In this embodiment of this application, an example in which the server uses a pre-trained language model as the code processing model is used for description. In some embodiments, the server pre-trains the language model again based on the large amount of training data, to obtain the code processing model. The pre-training is unsupervised training, and includes a plurality of rounds of training processes. Before training, the server performs text segmentation on the large amount of training data to obtain a plurality of words, and converts the plurality of words into vectors. Each vector indicates information such as a location of a corresponding word in the large amount of training data and a correlation between the corresponding word and another word. In any round of training process, the server invokes the language model to calculate the vectors, to obtain a plurality of next words of each word and a corresponding probability. If the probability meets a training stop condition, the corresponding language model is used as the code processing model. If the probability does not meet the training stop condition, back propagation training is performed on the language model based on the probability, to adjust a parameter in the language model; and then a next round of training is performed. The training stop condition may be that the probability is within a specific value range, or a quantity of rounds of model training reaches a preset quantity of rounds. This is not limited in this embodiment of this application. The server pre-trains the language model by using the large amount of training data, so that knowledge learned by a word processing model can better match a field corresponding to a code processing task, and accuracy of code processing is higher.

[00117] 901: The server obtains a first training dataset, where the first training dataset includes a plurality of groups of first sample data and first label, the first sample data includes one group of first code and modification suggestion information, and the first label in a same group as the first sample data includes one group of modification intent and code modification location corresponding to the first sample data.

[00118] The first sample data is input data of the code processing model, the first label is a real

label corresponding to the first sample data, and the first sample data may be manually collected or manually labeled. In this embodiment of this application, the server locally reads a pre-downloaded first training dataset, or obtains a first training dataset from an open source library. In this embodiment of this application, an example in which the first sample data is manually labeled to obtain the first training dataset is used for description. In some embodiments, the server records related information in a historical code processing process, to obtain the first training dataset. This is not limited in this embodiment of this application.

[00119] In some embodiments, the server divides obtained data into the first training dataset and a first test dataset based on a preset proportion, which are briefly referred to as a training set and a test set. The training set is used to train a model, and the test set is used to check a model training result. The model can be further trained based on the model training result, to improve model performance.

[00120] A process of training the code processing model includes a plurality of rounds of training. The following steps 902 to 904 describe one round of training process.

[00121] 902: In each round of training, the server inputs the first sample data into the code processing model, to obtain first output data.

[00122] The first output data is a plurality of combinations and a probability of each combination, each combination includes one group of modification intent and code modification location, and the probability indicates a probability that an output is a corresponding combination when input is the first sample data. In some embodiments, the server fine-tunes the code processing model through each round of training based on the first training dataset, so that a fine-tuned model can be used in a preset field. Before training, the server performs text segmentation on the first sample data to obtain a plurality of words, and converts the plurality of words into vectors. Each vector indicates information such as a location of a corresponding word in the first sample data and a correlation between the corresponding word and another word. In each round of training, the server invokes the code processing model to calculate the vectors corresponding to the first sample data, to obtain the first output data.

[00123] In this embodiment of this application, an example in which the first output data is the plurality of combinations and the probability of each combination is used for description. In some embodiments, the first output data is one group of modification intent and modification location.

[00124] 903: The server obtains a first loss function value based on the first output data and the first label.

[00125] The first output data is a modification intent and a code modification location that are predicted by the code processing model, the first label is a real modification intent and a real code modification location, and the first loss function value indicates prediction accuracy of the code

processing model. In this embodiment of this application, the server obtains a probability distribution of the plurality of combinations of the first output data and a probability distribution of real labels of the first sample data, and transforms the two probability distributions according to a preset function, to obtain the first loss function value.

[00126] In this embodiment of this application, an example in which the first output data is the plurality of combinations and the probability of each combination is used for description. In some embodiments, the first output data is one group of modification intent and modification location. In this case, the server uses an error between the first output data and the first label as the first loss function value.

[00127] 904: If this round of training does not meet the training stop condition, the server adjusts a parameter in the code processing model based on the first loss function value; and if this round of training meets the training stop condition, the server outputs the model as the code processing model.

[00128] The adjusted parameter may be some parameters in the code processing model, or may be a newly added parameter in the code processing model. This is not limited in this embodiment of this application. The first loss function value indicates a difference between a predicted probability distribution and a real probability distribution, that is, a difference between the probability distribution of the plurality of combinations of the first output data and the probability distribution of the real labels of the first sample data. If this round of training does not meet the training stop condition, the server adjusts the parameter in the code processing model based on the first loss function value, and then performs a next round of training, so that the difference indicated by the first loss function value is smaller, until the training stop condition is met. The training stop condition may be that the first loss function value is within the specific value range, or may be that the quantity of rounds of model training reaches the preset quantity of rounds. This is not limited in this embodiment of this application. If this round of training meets the training stop condition, the model is output as the code processing model.

[00129] In this embodiment of this application, an example in which the first loss function value indicates the difference between the predicted probability distribution and the real probability distribution is used for description. In some embodiments, the first loss function value indicates an error between a prediction label of the code processing model and the first label.

[00130] In some embodiments, the server tunes the code processing model through model supervised fine-tuning (supervised fine-tuning, SFT) and prompt engineering, so that the code processing model has an end-to-end processing capability, and can generate both the modification intent and the code modification location.

[00131] Steps 902 to 904 are a possible implementation of training the code processing model

based on the first training dataset. In this implementation, the code processing model is an LLM, training can be performed on an existing language model, and the model does not need to be trained from the beginning, which can save computation time and compute resources. In some embodiments, the code processing model may alternatively be another model having a natural language processing capability or a powerful classification capability.

[00132] The code processing model show

[00133] n in FIG. 9 is obtained through training by using a large amount of basic training data and training data in the specific field, has a powerful natural language processing capability for the specific field, and can accurately output both the modification intent and the code modification location based on the first code and the modification suggestion information, thereby improving code processing efficiency.

[00134] FIG. 10 is a schematic flowchart of performing code processing based on an intent recognition model and a locating model. FIG. 10 includes three processes. One process is obtaining first code and modification suggestion information, inputting the first code and the modification suggestion information into the intent recognition model to obtain a modification intent, inputting the first code, the modification suggestion information, and the modification intent into the locating model to obtain a code modification location, and then processing the first code based on the modification intent and the code modification location. The second process is training the intent recognition model based on a training dataset. The last process is training the locating model based on the training data set. The following further describes the three processes based on FIG. 11A and FIG. 11B to FIG. 13.

[00135] FIG. 11A and FIG. 11B are a flowchart of a code processing method according to an embodiment of this application. With reference to FIG. 11A and FIG. 11B, the following describes in more detail the code processing method implemented based on the intent recognition model and the locating model by using an example in which a code development client is the code development client, the code development client obtains the first code, and a server performs the foregoing code processing method. The method includes the following steps.

[00136] 1100: The code development client displays a code processing interface, where the code processing interface is used to provide a code processing function.

[00137] 1101: The code development client obtains, in response to an operation performed on the code processing function in the code processing interface, the first code obtained through development.

[00138] 1102: The code development client sends a code processing instruction to the server in response to a commit operation on the first code, where the code processing instruction carries the first code.

[00139] 1103: The server receives the code processing instruction.

[00140] 1104: The server obtains the first code based on the code processing instruction.

[00141] 1105: The server obtains modification suggestion information for the first code based on the first code by using a suggestion obtaining model, and returns the modification suggestion information for the first code to the code development client, where the modification suggestion information includes a descriptive text and a to-be-processed code segment, the descriptive text indicates a modification intent for the first code based on the code segment, and the modification intent includes deleting the code segment from the first code, adding the code segment to the first code, and replacing a code segment in the first code with the code segment.

[00142] 1106: The code development client displays the modification suggestion information for the first code; and sends an instruction of continuing processing to the server in response to a trigger operation for the modification suggestion information, or stops a code processing process in response to a trigger operation for the modification suggestion information.

[00143] Steps 1100 to 1106 are similar to steps 700 to 706. Details are not described herein again in this embodiment of this application.

[00144] 1107: If the instruction of continuing processing is received, the server obtains the modification intent for the first code based on the first code and the modification suggestion information for the first code by using the intent recognition model.

[00145] In this embodiment of this application, if the instruction of continuing processing is received, the server concatenates the first code and the modification suggestion information in a preset sequence, invokes the intent recognition model, and processes concatenated first code and modification suggestion information, to obtain the modification intent. The preset sequence is consistent with a concatenation sequence of the first code and the modification suggestion information in data for training the intent recognition model. The intent recognition model is used to obtain the modification intent, to be specific, determine the modification intent is adding, deleting, or modifying. Inputs of the model are the first code and the modification suggestion information, and an output is one of the three types of the modification intent: adding, deleting, or modifying. The foregoing process of obtaining the modification intent may be: performing text segmentation on the concatenated first code and modification suggestion information to obtain a plurality of words, converting the plurality of words into a plurality of vectors, and invoking the intent recognition model to calculate the plurality of vectors to obtain the modification intent.

[00146] For example, the server concatenates the code and the modification suggestion information in a sequence in which the code is before the modification suggestion information, invokes the intent recognition model, and analyzes the code with line numbers and the modification suggestion information for the code, to obtain the modification intent.

[00147] In this embodiment of this application, an example in which the intent recognition model directly outputs one modification intent is used for description. In some embodiments, the intent recognition model first obtains a plurality of modification intents and a probability of each modification intent, then selects one modification intent from the plurality of modification intents, and outputs the modification intent. Selecting one modification intent from the plurality of modification intents may be selecting a modification intent with the highest probability, or may be randomly sampling one modification intent from the plurality of modification intents whose probabilities are greater than a probability threshold. The probability indicates a probability that the output is the corresponding modification intent when the inputs are the first code and the corresponding modification suggestion information.

[00148] In the foregoing embodiments, an example in which the intent recognition model outputs one modification intent is used for description. In some other embodiments, the intent recognition model outputs the plurality of modification intents and the probability corresponding to each modification intent. The server selects one modification intent from the plurality of modification intents based on the probabilities, and continues to perform a subsequent code processing process based on the modification intent. This is not limited in this embodiment of this application.

[00149] 1108: The server obtains, based on the first code, the modification suggestion information for the first code, and the modification intent, a code modification location, in the first code, that corresponds to the modification suggestion information.

[00150] In this embodiment of this application, the server concatenates the first code, the modification suggestion information, and the modification intent in a preset sequence, invokes the locating model, and processes concatenated first code, modification suggestion information, and modification intent, to obtain the code modification location corresponding to the modification suggestion information. The locating model is used to obtain the code modification location, in other words, determine a location, in the first code, at which modification needs to be performed. Inputs of the locating model are the first code, the modification suggestion information, and the modification intent, and an output is the code modification location. A process of obtaining the code modification location may be: performing text segmentation on the concatenated first code, modification suggestion information, and modification intent to obtain a plurality of words, converting the plurality of words into a plurality of vectors, and invoking the locating model to calculate the plurality of vectors, to obtain the code modification location.

[00151] For example, a line number indicates a location of code. The server concatenates the code, the modification suggestion information, and the modification intent in a sequence of the code, the modification suggestion information, and the modification intent; invokes the locating

model; and analyzes the code with line numbers, the modification suggestion information, and the modification intent, to obtain the code modification location, that is, the line number.

[00152] In this embodiment of this application, an example in which the output of the locating model is one code modification location is used for description. In some embodiments, the locating model first obtains a plurality of code modification locations and a probability of each code modification location by analyzing the first code, the modification suggestion information, and the modification intent; selects one code modification location from the plurality of code modification locations; and outputs the code modification location. Selecting one code modification location from the plurality of code modification locations may be selecting a code modification location with the highest probability, or may be randomly sampling one code modification location from the plurality of code modification locations whose probabilities are greater than a probability threshold. The probability indicates a probability that the output is the corresponding code modification location when the inputs are the first code, the corresponding modification suggestion information, and the corresponding modification intent.

[00153] In the foregoing embodiments, an example in which the locating model outputs one code modification location is used for description. In some other embodiments, the locating model outputs the plurality of code modification locations and the probability corresponding to each code modification location. The server selects one code modification location from the plurality of code modification locations based on the probabilities, and continues to perform a subsequent code processing process based on the code modification location. This is not limited in this embodiment of this application.

[00154] 1109: The server sends the modification intent and the code modification location of the modification suggestion information for the first code to the code development client.

[00155] 1110: The code development client performs, at the code modification location in the first code based on the modification intent and the code modification location, processing corresponding to the modification intent based on the code segment, to obtain second code.

[00156] 1111: Display the second code in a preview window of the code processing interface.

[00157] 1112: If the code development client receives an approval instruction for the second code displayed in the preview window, the code development client replaces the first code with the second code.

[00158] Steps 1109 to 1112 are similar to steps 708 to 711. Details are not described herein again in this embodiment of this application.

[00159] In the code processing method shown in FIG. 10, the modification intent is obtained by using the intent recognition model. The intent recognition model may be a plurality types of natural language processing models. FIG. 12 is a flowchart of an intent recognition model training

method according to an embodiment of this application. With reference to FIG. 12, the following uses an example in which a server performs the model training method for description. The method includes the following steps.

[00160] 1201: The server obtains a second training dataset, where the second training dataset includes a plurality of groups of second sample data and second label, the second sample data includes one group of first code and modification suggestion information, and the second label in a same group as the second sample data includes one modification intent corresponding to the second sample data.

[00161] The second sample data is input data of the intent recognition model, and the second label is a real label corresponding to the second sample data. A process of obtaining the second training dataset in step 1201 is similar to a process of obtaining the first training dataset in step 901. Details are not described again in this embodiment of this application.

[00162] A process of training the intent recognition model includes a plurality of rounds of training. The following steps 1202 to 1204 describe one round of training process.

[00163] 1202: In each round of training, the server inputs the second sample data into the intent recognition model, to obtain second output data.

[00164] The intent recognition model is a model selected based on a code processing task. Before training, the server performs text segmentation on the second sample data to obtain a plurality of words, and converts the plurality of words into vectors. Each vector indicates information such as a location of a corresponding word in the second sample data and a correlation between the corresponding word and another word. In each round of training, the server invokes the intent recognition model to calculate the vectors corresponding to the second sample data, to obtain the second output data, that is, a modification intent predicted by the intent recognition model.

[00165] In some embodiments, the intent recognition model is a recurrent neural network (recurrent neural network, RNN) model. In one round of training process, forward propagation calculation is performed on the vectors corresponding to the second sample data in the RNN model, to obtain the second output data. The vectors corresponding to the second sample data are calculated by a neural network layer such as a recurrent layer, a hidden layer, and a fully connected layer in the RNN model, to obtain the second output data.

[00166] In some embodiments, the intent recognition model is a transformer model. The transformer model includes a plurality of encoder (encoder) modules and decoder (decoder) modules. The encoder module is configured to encode input data, to convert the input data into intermediate data. The decoder module is configured to decode the intermediate data, to convert the intermediate data into output data. A structure of the encoder module is the same as that of the

decoder module, but parameters of the two modules may be different. Each encoder module or decoder module includes a self-attention (self-attention, SA) layer and a feedforward neural network layer (feedforward neural network, FFNN). The encoder module is used as an example. When encoding a word in the input data, the SA layer is configured to enable encoded word to include information about another word in the input data, in other words, include a relationship between the word and the another word. The FFNN layer is a forward network layer, and is used to perform further feature extraction on an output of the SA layer. In one round of training process, the encoder module is invoked to process the vectors corresponding to the second sample data, to obtain intermediate data, and the decoder module is invoked to process the intermediate data, to obtain the second output data.

[00167] In some embodiments, the intent recognition model is a Naive Bayes classification model. In one round of training process, the Naive Bayes classification model is invoked to calculate the vectors corresponding to the second sample data, to obtain at least one modification intent and a probability corresponding to each modification intent. The probability indicates a probability that an output is a corresponding modification intent when an input is the second sample data, and a modification intent with the highest probability in the at least one modification intent is used as the second output data.

[00168] 1203: The server obtains a second loss function value based on the second output data and the second label.

[00169] The second output data is the modification intent predicted by the intent recognition model, the second label is a real modification intent, and the second loss function value indicates an error between the second output data and the second label. The server invokes a preset loss function, and performs calculation based on the second output data and the second label, to obtain the second loss function value.

[00170] 1204: If this round of training does not meet a training stop condition, the server adjusts a parameter in the intent recognition model based on the second loss function value; and if this round of training meets the training stop condition, the server outputs the model as the intent recognition model.

[00171] In this embodiment of this application, if this round of training does not meet the training stop condition, the server calculates an error of each parameter in the model based on the second loss function value. The parameter in the intent recognition model is adjusted based on the error, and then a next round of training is performed, so that the second loss function value is smaller until the training stop condition is met. The training stop condition may be that the second loss function value is within a specific value range, or may be that a quantity of rounds of model training reaches a preset quantity of rounds. This is not limited in this embodiment of this

application. If this round of training meets the training stop condition, the model is output as the intent recognition model.

[00172] Steps 1202 to 1204 are a possible implementation of training the intent recognition model based on the second training dataset. In this implementation, a plurality of models that can be used as the intent recognition model are provided. The model mentioned in this implementation is merely an example, and the intent recognition model may alternatively be another natural language processing model.

[00173] In the code processing method shown in FIG. 10, the code modification location is obtained by using the locating model. The locating model may be a plurality of natural language processing models. FIG. 13 is a flowchart of a locating model training method according to an embodiment of this application. With reference to FIG. 13, the following uses an example in which a server performs the model training method for description. The method includes the following steps.

[00174] 1301: The server obtains a third training dataset, where the third training dataset includes a plurality of groups of third sample data and third label, the third sample data includes one group of first code, modification suggestion information, and modification intent, and the third label in a same group as the third sample data includes one code modification location corresponding to the third sample data.

[00175] The third sample data is input data of the locating model, and the third label is a real label corresponding to the third sample data. A process of obtaining the third training dataset in step 1301 is similar to the process of obtaining the first training dataset in step 901. Details are not described again in this embodiment of this application.

[00176] A process of training the locating model includes a plurality of rounds of training. The following steps 1302 to 1304 describe one round of training process.

[00177] 1302: In each round of training process, the server inputs the third sample data into the locating model, to obtain third output data.

[00178] 1303: The server obtains a third loss function value based on the third output data and the third label.

[00179] 1304: If this round of training does not meet a training stop condition, the server adjusts a parameter in the locating model based on the third loss function value; and if this round of training meets the training stop condition, the server outputs the model as the locating model.

[00180] Steps 1302 to 1304 are a possible implementation of training the locating model based on the third training dataset; and in this implementation, a plurality of models that can be used as the locating model are provided. The model mentioned in this implementation is merely an example, and the locating model may alternatively be another natural language processing model.

[00181] The locating model training method in steps 1301 to 1304 is similar to the intent recognition model training method in steps 1201 to 1204. Details are not described again in this embodiment of this application.

[00182] According to the code processing method described in FIG. 11A and FIG. 11B to FIG. 13, two models are used to separately determine the modification intent and the code modification location. Objectives of the models are relatively clear, and prediction accuracy is relatively high.

[00183] The method in embodiments of this application is described above. An apparatus in embodiments of this application is described below. It should be understood that the apparatus described below has any function of the compute device in the foregoing method. The foregoing describes in detail the code processing method provided in embodiments of this application and the model training method used in the code processing method with reference to FIG. 6 to FIG. 13. Based on a same inventive concept, the following describes a code processing apparatus according to an embodiment of this application with reference to FIG. 14. It should be understood that the technical features described in the method embodiments are also applied to the following apparatus embodiment.

[00184] Refer to FIG. 14. An embodiment of this application provides a code processing apparatus. The apparatus includes:

a code obtaining module 1401, configured to obtain first code obtained through development;

a modification suggestion information obtaining module 1402, configured to analyze the obtained first code in response to a code analysis instruction, to obtain modification suggestion information for the first code, where the modification suggestion information includes a descriptive text and a to-be-processed code segment, the descriptive text indicates a modification intent for the first code based on the code segment, and the modification intent includes deleting the code segment from the first code, adding the code segment to the first code, and replacing a code segment in the first code with the code segment;

a suggestion recognition module 1403, configured to perform analysis based on the modification suggestion information obtained through analysis and the first code, to obtain the modification intent and a code modification location, in the first code, that corresponds to the modification suggestion information; and

a code processing module 1404, configured to perform, at the code modification location in the first code based on the modification intent and the code modification location, processing corresponding to the modification intent based on the code segment.

[00185] In some embodiments, the suggestion recognition module 1403 is configured to:

in response to a trigger operation for the modification suggestion information, perform

analysis based on the modification suggestion information and the first code, to obtain the modification intent and the code modification location.

[00186] In some embodiments, the code processing module 1404 includes:

a second code obtaining unit, configured to perform, at the code modification location in the first code based on the modification intent and the code modification location, processing corresponding to the modification intent based on the code segment, to obtain second code;

a display unit, configured to display the second code in a preview window; and

a replacement unit, configured to: if an approval instruction is received, replace the first code with the second code.

[00187] In some embodiments, the second code obtaining unit is configured to:

if the modification intent is adding, insert the code segment in the modification suggestion information after code at the code modification location in the first code, to obtain the second code;

if the modification intent is deleting, delete code at the code modification location from the first code, to obtain the second code; or

if the modification intent is modifying, delete code at the code modification location from the first code, and insert the code segment in the modification suggestion information at the code modification location, to obtain the second code.

[00188] In some embodiments, the code processing apparatus further includes:

a code processing model training module, configured to obtain a first training dataset, where the first training dataset includes a plurality of groups of first sample data and first label, the first sample data includes one group of first code and modification suggestion information, and the first label includes one group of modification intent and code modification location corresponding to the first sample data; and train the code processing model based on the first training dataset.

[00189] In some embodiments, the code processing apparatus further includes:

an intent recognition model training module, configured to obtain a second training dataset, where the second training dataset includes a plurality of groups of second sample data and second label, the second sample data includes one group of first code and modification suggestion information, and the second label includes one modification intent corresponding to the second sample data; and train the intent recognition model based on the second training dataset.

[00190] In some embodiments, the code processing apparatus further includes:

a locating model training module, configured to obtain a third training dataset, where the third training dataset includes a plurality of groups of third sample data and third label, the third sample data includes one group of first code, modification suggestion information, and

modification intent, and the third label includes one code modification location corresponding to the third sample data; and train the locating model based on the third training dataset.

[00191] It should be understood that the code processing apparatus corresponds to the code development client in the foregoing method embodiments. The modules in the apparatus and the foregoing other operations and/or functions are separately intended to implement various steps and methods implemented by the code development client in the method embodiments. For specific details, refer to the foregoing method embodiments. For brevity, details are not described herein again.

[00192] The code obtaining module 1401, the modification suggestion information obtaining module 1402, the suggestion recognition module 1403, and the code processing module 1404 may all be implemented through software, or may be implemented through hardware. For example, the following uses the code obtaining module 1401 as an example to describe an implementation of the code obtaining module 1401. Similarly, for implementations of the modification suggestion information obtaining module 1402, the suggestion recognition module 1403, and the code processing module 1404, refer to the implementation of the code obtaining module 1401.

[00193] As an example of a software functional unit, the code obtaining module 1401 may include code run on a computing instance. The computing instance may include at least one of a physical host (compute device), a virtual machine, and a container. Further, there may be one or more computing instances. For example, the code obtaining module 1401 may include code running on a plurality of hosts/virtual machines/containers. It should be noted that, the plurality of hosts/virtual machines/containers configured to run the code may be distributed in a same region, or may be distributed in different regions. Further, the plurality of hosts/virtual machines/containers configured to run the code may be distributed in a same AZ, or may be distributed in different AZs. Each AZ includes one data center or a plurality of data centers with close geographical locations. Usually, one region may include a plurality of AZs.

[00194] Similarly, the plurality of hosts/virtual machines/containers configured to run the code may be distributed in a same VPC, or may be distributed in a plurality of VPCs. Usually, one VPC is set in one region. A communication gateway needs to be set in each VPC for communication between two VPCs in a same region or between VPCs in different regions. Interconnection between the VPCs is implemented through the communication gateway.

[00195] As an example of a hardware functional unit, the code obtaining module 1401 may include at least one compute device, for example, a server. Alternatively, the code obtaining module 1401 may be a device implemented through an ASIC, a PLD, or the like. The PLD may be implemented through a CPLD, an FPGA, GAL, or any combination thereof.

[00196] The plurality of compute devices included in the code obtaining module 1401 may be

distributed in a same region, or may be distributed in different regions. The plurality of compute devices included in the code obtaining module 1401 may be distributed in a same AZ, or may be distributed in different AZs. Similarly, the plurality of compute devices included in the code obtaining module 1401 may be distributed in a same VPC, or may be distributed in a plurality of VPCs. The plurality of compute devices may be any combination of compute devices such as a server, an ASIC, a PLD, a CPLD, an FPGA, and GAL.

[00197] It should be noted that, in another embodiment, the code obtaining module 1401 may be configured to perform any step in the code processing method, the modification suggestion information obtaining module 1402 may be configured to perform any step in the code processing method, the suggestion recognition module 1403 may be configured to perform any step in the code processing method, and the code processing module 1404 may be configured to perform any step in the code processing method. Steps that are implemented by the code obtaining module 1401, the modification suggestion information obtaining module 1402, the suggestion recognition module 1403, and the code processing module 1404 may be specified according to requirements. The code obtaining module 1401, the modification suggestion information obtaining module 1402, the suggestion recognition module 1403, and the code processing module 1404 respectively implement different steps in the code processing method, to implement all functions of the code processing apparatus.

[00198] Embodiments of this application further provide a computer-readable storage medium. The computer-readable storage medium may be any usable medium accessible by a compute device, or a data storage device, such as a data center, including one or more usable media. The usable medium may be a magnetic medium (for example, a floppy disk, a hard disk, or a magnetic tape), an optical medium (for example, a high-density digital video disc (digital video disc, DVD)), a semiconductor medium (for example, a solid-state drive), or the like. The computer-readable storage medium includes computer program instructions. When the computer program instructions are executed by a compute device cluster, the compute device cluster performs the code processing method.

[00199] Embodiments of this application further provide a computer program product including instructions, and the computer program product is software or a program product that can run on a compute device or that is stored in any available medium. When the computer program product runs on at least one compute device, the at least one compute device is caused to perform the code processing method.

[00200] In addition, embodiments of this application further provide an apparatus. The apparatus may be specifically a chip, a component, or a module. The apparatus may include a processor and a memory that are connected to each other. The memory is configured to store

computer-executable instructions. When the apparatus runs, the processor may execute the computer-executable instructions stored in the memory, to cause the chip to perform the code processing method in the foregoing method embodiments.

[00201] The apparatus, the compute device cluster, the computer-readable storage medium, the computer program product, or the chip provided in this embodiment is configured to perform the corresponding method provided above. Therefore, for beneficial effects that can be achieved by the apparatus, the compute device cluster, the computer-readable storage medium, the computer program product, or the chip, refer to the beneficial effects in the corresponding method provided above. Details are not described herein again.

[00202] Based on the foregoing descriptions of the implementations, persons skilled in the art may understand that for the purpose of convenient and brief description, division into the foregoing functional modules is merely used as an example for illustration. During actual application, the foregoing functions can be allocated to different functional modules for implementation according to requirements. In other words, an inner structure of an apparatus is divided into different functional modules to implement all or a part of the functions described above. In addition, this embodiment belongs to a same concept as the code processing method embodiments provided in the foregoing embodiments. For a specific implementation process, refer to the method embodiments. Details are not described herein again.

[00203] In the several embodiments provided in this application, it should be understood that the disclosed apparatus and method may be implemented in other manners. For example, the described apparatus embodiment is merely an example. For example, the module or division into the units is merely logical function division and may be other division in actual implementation. For example, a plurality of units or components may be combined or integrated into another apparatus, or some features may be ignored or not performed. In addition, the displayed or discussed mutual couplings or direct couplings or communication connections may be implemented through some interfaces. The indirect couplings or communication connections between the apparatuses or units may be implemented in electronic, mechanical, or other forms.

[00204] The units described as separate parts may or may not be physically separate, and parts displayed as units may be one or more physical units, may be located in one place, or may be distributed on different places. Some or all of the units may be selected according to actual requirements to achieve the objectives of the solutions of embodiments.

[00205] In addition, functional units in embodiments of this application may be integrated into one processing unit, or each of the units may exist alone physically, or two or more units may be integrated into one unit. The integrated unit may be implemented in a form of hardware, or may be implemented in a form of a software functional unit.

[00206] When the integrated unit is implemented in the form of a software functional unit and sold or used as an independent product, the integrated unit may be stored in a readable storage medium. Based on such an understanding, the technical solutions in embodiments of this application essentially, or the part contributing to the conventional technology, or all or some of the technical solutions may be embodied in a form of a software product. The software product is stored in a storage medium, and includes several instructions for instructing a device (which may be a single-chip microcomputer, a chip, or the like) or a processor (processor) to perform all or some of the steps in the method in embodiments of this application. The foregoing storage medium includes any medium that can store program code, such as a USB flash drive, a removable hard disk, a ROM, a RAM, a magnetic disk, or an optical disc.

[00207] In descriptions of this application, unless otherwise specified, "/" means "or". For example, A/B may indicate A or B. A term "and/or" in this specification describes only an association relationship between associated objects and indicates that there may be three relationships. For example, A and/or B may represent the following three cases: Only A exists, both A and B exist, and only B exists. In addition, "at least one" means one or more, and "a plurality of" means two or more. Terms such as "first" and "second" do not limit a quantity and an execution sequence, and the terms such as "first" and "second" do not indicate a definite difference.

[00208] In this application, terms such as "example" and "for example" are used for giving an example, an illustration, or a description. Any embodiment or design scheme described as an "example" or "for example" in this application should not be explained as being more preferred or having more advantages than another embodiment or design scheme. To be precise, use of the words such as "example" or "for example" is intended to present a relative concept in a specific manner.

[00209] It should be noted that information (including but not limited to user equipment information, user personal information, and the like), data (including but not limited to data for analysis, data for storage, data for display, and the like), and signals in this application are all authorized by users or fully authorized by all parties, and collection, use, and processing of related data need to comply with related laws, regulations, and standards of related countries and regions. For example, the code in this application is obtained under full authorization.

[00210] All of the foregoing technical solutions may form optional embodiments of this disclosure through any combination. Details are not described herein again.

[00211] Finally, it should be noted that the foregoing embodiments are merely intended for describing the technical solutions of this application, but not for limiting this application. Although this application is described in detail with reference to the foregoing embodiments, persons of ordinary skill in the art should understand that modifications may still be made to the technical

solutions described in the foregoing embodiments or equivalent replacements may be made to some technical features, and such modifications and replacements do not cause the essence of the corresponding technical solutions to depart from the protection scope of technical solutions of embodiments of this application.

[00212] Persons of ordinary skill in the art may understand that all or some of the steps of the embodiments may be implemented through hardware or a program instructing related hardware. The program may be stored in a computer-readable storage medium. The storage medium may be a read-only memory, a magnetic disk, an optical disc, or the like.

[00213] The foregoing descriptions are merely example embodiments of this application, but are not intended to limit this application. Any modification, equivalent replacement, improvement, or the like made without departing from the spirit and principle of this application shall fall within the protection scope of this application.

CLAIMS

1. A code processing method, wherein the method comprises:

obtaining, by a code development client, first code obtained through development;

analyzing, by the code development client, the obtained first code in response to a code analysis instruction, to obtain modification suggestion information for the first code, wherein the modification suggestion information comprises a descriptive text and a to-be-processed code segment, the descriptive text indicates a modification intent for the first code based on the code segment, and the modification intent comprises deleting the code segment from the first code, adding the code segment to the first code, and replacing a code segment in the first code with the code segment;

performing, by the code development client, analysis based on the modification suggestion information obtained through analysis and the first code, to obtain the modification intent and a code modification location, in the first code, that corresponds to the modification suggestion information; and

performing, by the code development client at the code modification location in the first code based on the modification intent and the code modification location, processing corresponding to the modification intent based on the code segment.

2. The method according to claim 1, wherein performing, by the code development client, analysis based on the modification suggestion information obtained through analysis and the first code, to obtain the modification intent and the code modification location, in the first code, that corresponds to the modification suggestion information comprises:

in response to a trigger operation for the modification suggestion information, performing, by the code development client, analysis based on the modification suggestion information and the first code, to obtain the modification intent and the code modification location.

3. The method according to claim 1, wherein performing, by the code development client at the code modification location in the first code based on the modification intent and the code modification location, processing corresponding to the modification intent based on the code segment comprises:

performing, by the code development client at the code modification location in the first code based on the modification intent and the code modification location, processing corresponding to the modification intent based on the code segment, to obtain second code;

displaying, by the code development client, the second code in a preview window; and

if an approval instruction is received, replacing, by the code development client, the first code with the second code.

4. The method according to claim 3, wherein performing, by the code development client at the code modification location in the first code based on the modification intent and the code modification location, processing corresponding to the modification intent based on the code segment, to obtain the second code comprises:

if the modification intent is adding, inserting, by the code development client, the code segment in the modification suggestion information after code at the code modification location in the first code, to obtain the second code;

if the modification intent is deleting, deleting, by the code development client, code at the code modification location from the first code, to obtain the second code; or

if the modification intent is modifying, deleting, by the code development client, code at the code modification location from the first code, and inserting the code segment in the modification suggestion information at the code modification location, to obtain the second code.

5. The method according to any one of claims 1 to 4, wherein the modification intent and the code modification location are obtained by using a code processing model, and a training process of the code processing model comprises:

obtaining a first training dataset, wherein the first training dataset comprises a plurality of groups of first sample data and first label, the first sample data comprises one group of first code and modification suggestion information, and the first label comprises one group of modification intent and code modification location corresponding to the first sample data; and

training the code processing model based on the first training dataset.

6. The method according to any one of claims 1 to 4, wherein the modification intent is obtained by using an intent recognition model, and a training process of the intent recognition model comprises:

obtaining a second training dataset, wherein the second training dataset comprises a plurality of groups of second sample data and second label, the second sample data comprises one group of first code and modification suggestion information, and the second label comprises one modification intent corresponding to the second sample data; and

training the intent recognition model based on the second training dataset.

7. The method according to any one of claims 1 to 4, wherein the code modification location is obtained by using a locating model, and a training process of the locating model comprises:

obtaining a third training dataset, wherein the third training dataset comprises a plurality of groups of third sample data and third label, the third sample data comprises one group of first code, modification suggestion information, and modification intent, and the third label comprises one code modification location corresponding to the third sample data; and

training the locating model based on the third training dataset.

8. A code processing apparatus, wherein the apparatus comprises:

a code obtaining module, configured to obtain first code obtained through development;

a modification suggestion information obtaining module, configured to analyze the obtained first code in response to a code analysis instruction, to obtain modification suggestion information for the first code, wherein the modification suggestion information comprises a descriptive text and a to-be-processed code segment, the descriptive text indicates a modification intent for the first code based on the code segment, and the modification intent comprises deleting the code segment from the first code, adding the code segment to the first code, and replacing a code segment in the first code with the code segment;

a suggestion recognition module, configured to perform analysis based on the modification suggestion information obtained through analysis and the first code, to obtain the modification intent and a code modification location, in the first code, that corresponds to the modification suggestion information; and

a code processing module, configured to perform, at the code modification location in the first code based on the modification intent and the code modification location, processing corresponding to the modification intent based on the code segment.

9. The apparatus according to claim 8, wherein the suggestion recognition module is configured to:

in response to a trigger operation for the modification suggestion information, perform analysis based on the modification suggestion information and the first code, to obtain the modification intent and the code modification location.

10. The apparatus according to claim 8, wherein the code processing module comprises:

a second code obtaining unit, configured to perform, at the code modification location in the first code based on the modification intent and the code modification location, processing corresponding to the modification intent based on the code segment, to obtain second code;

a display unit, configured to display the second code in a preview window; and

a replacement unit, configured to: if an approval instruction is received, replace the first code with the second code.

11. The apparatus according to claim 10, wherein the second code obtaining unit is configured to:

if the modification intent is adding, insert, by the code development client, the code segment in the modification suggestion information after code at the code modification location in the first code, to obtain the second code;

if the modification intent is deleting, delete, by the code development client, code at the code modification location from the first code, to obtain the second code; or

if the modification intent is modifying, delete, by the code development client, code at the code modification location from the first code, and insert the code segment in the modification suggestion information at the code modification location, to obtain the second code.

12. A compute device cluster, comprising at least one compute device, wherein each compute device comprises a processor and a memory; and

the processor of the at least one compute device is configured to execute instructions stored in the memory of the at least one compute device, so that the compute device cluster performs the method according to any one of claims 1 to 7.

13. A computer-readable storage medium, comprising computer program instructions, wherein when the computer program instructions are executed by a compute device cluster, the compute device cluster performs the method according to any one of claims 1 to 7.

14. A computer program product comprising instructions, wherein when the instructions are run by a compute device cluster, the compute device cluster is caused to perform the method according to any one of claims 1 to 7.